مجله کنترل
ISSN 2008-8345
جلد ۸، شماره ۲، تابستان ۱۳۹۳، صفحه ۲۰-۱۱

کنترل بینامبنای بازوی ربات با مدل‌سازی عصبی معکوس ماتریس ژاکوبین

فرزانه نادی[۱]، ولی درهمی[۲]، مهدی رضاییان[۳]

[۱] فارغ‌التحصیل کارشناسی ارشد مهندسی برق و کامپیوتر، گروه کامپیوتر، دانشگاه یزد، farzane.nadi@yahoo.com

[۲] دانشیار، دانشکدهٔ مهندسی برق و کامپیوتر، گروه کامپیوتر، دانشگاه یزد، vderhami@yazd.ac.ir

[۳] استادیار، دانشکدهٔ مهندسی برق و کامپیوتر، گروه کامپیوتر، دانشگاه یزد، mrezaeian@yazd.ac.ir

(تاریخ دریافت مقاله ۱۳۹۳/۱/۲۱، تاریخ پذیرش مقاله ۱۳۹۳/۵/۲۰)

چکیده: سیستم کنترل خودفرمان بینایی، به سیستمی اتلاق می‌شود که از اطلاعات بازخوردی دوربین برای کنترل ربات استفاده می‌کند؛ تا ربات، از نقاط شروع دلخواه به نقطه هدف برسد. راه‌های متنوعی از جمله کنترل با استفاده از مدل ربات، طراحی کنترلگر بصورت مستقیم، و استفاده از ماتریس ژاکوبین در این زمینه مطرح شده است. اما، از آنجا که در بسیاری از مواقع، مدلی از ربات دردسترس نیست و یا بدست آوردن آن کاری دشوار و زمانبر خواهد بود، فرمان کنترل با استفاده از ماتریس ژاکوبین تولید می‌شود. در اینجا، با استفاده از شبکه‌های عصبی مصنوعی معکوس ماتریس ژاکوبین تقریب زده می‌شود؛ تا مستقیماً در قانون کنترل موردنظر استفاده شود. به ازای هر کدام از درجات آزادی بازوی ربات، یک شبکه عصبی دو لایه با ساختار پیشرو درنظر گرفته شده است. فاصله مجری‌نهایی با هدف و مختصات مفصل شانه (در دو بعد x و y)، ورودی‌های این شبکه‌ها، و خروجی آنها، بیان کننده میزان تغییرات ویژگی‌ها به تغییر در مقادیر متغیرهای مفصل ربات (المانهای معکوس ماتریس ژاکوبین) می‌باشد. روش ارائه شده بر روی بازوی ربات واقعی پیاده‌سازی شده است. نتایج آزمایشات حاکی از موفقیت روش پیشنهادی در رساندن مجری‌نهایی به نقاط هدف مختلف در محیط کاری با دقت مناسب است.

کلمات کلیدی: کنترل خودفرمان بینایی، ماتریس ژاکوبین، بازوی ربات، شبکه‌های عصبی مصنوعی.

Vision Based Robot Manipulator Control with Neural Modeling of Jacobian Matrix

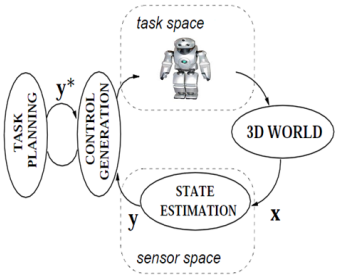
Farzane Nadi, Vali Derhami, Mehdi Rezaeian

Abstract: Visual servoing system is a system to control a robot by visual feedback so that robot drives from any arbitrary start position to target positions. Various ways, including control by using model of the robot, designing controller directly, and using Jacobian matrix have been studied. Since there is not access to model of robot and obtaining a model of robot would be difficult and time consuming, in many cases, the control law is obtained using Jacobian matrix. In this paper, inverse of Jacobian matrix is approximated by artificial neural networks. The approximated neural models are used in control law directly. For each degree of freedom of the robot manipulator, a two-layer feedforward neural network is considered. The distance between end-effector and target, and the shoulder joint coordinates (along the x-axis and y-axis) are the inputs of each of the networks and the outputs are the fraction of the related robot joint changes to the image features changes (the elements of the inverse of Jacobian matrix). The proposed method has been implemented on a real robot manipulator .The experimental results show that the proposed control system can move the end-effector to different target positions in workspace with good accuracy.

Keywords: Visual servoing, Jacobian matrix, robot manipulator, artificial neural network.

۱- مقدمه

کنترل خودفرمان بینایی[1] تکنیکی است که از داده‌های بازخوردی استخراج شده توسط حسگرهای بصری استفاده می‌کند تا حرکت ربات را کنترل کند [۱]. این روش، راه‌حلی برای کنترل ربات در محیط‌های ساخت یافته می‌باشد و شامل تکنیک‌های مختلفی از جمله پردازش تصویر، بینایی ماشین و تئوری کنترل می‌باشد [۲]–[۴]. از اجزای مهم سیستم‌های رباتیکی جمع‌آوری، پردازش و تفسیر داده‌های حسگرها می‌باشد [۴] که حسگر مورد استفاده در این روش دوربین می‌باشد؛ چرا که دوربین، اطلاعاتی با دقت مناسب، منعطف، و در عین‌حال کم هزینه را دردسترس قرار می‌دهد و می‌تواند در یک مکان بصورت ثابت نصب شود[2] و یا روی بازوی ربات قرار بگیرد[3] [۱], [۵]. پردازش اصلی در کنترل خودفرمان بینایی، همانطور که در شکل ۱ مشخص می‌باشد، به گونه‌ای است که واحد تخمین حالت با استفاده از بردار حالت سیستم (X)، بردار اندازه (Y) که همان مقادیر ویژگی‌ها می‌باشد، را محاسبه می‌کند. سپس واحد کنترل با استفاده از بردار اندازه و بردار اندازه مطلوب (Y*) حرکت نسبی اقلیدسی ربات را به عنوان ورودی (u) به ربات می‌دهد و درنهایت حرکت ربات در فضای سه بعدی اتفاق می‌افتد [۶].



شکل ۱: پردازش اصلی در کنترل خودفرمان بینایی [۶]

کنترل خودفرمان بینایی از دیدگاه‌های مختلف دسته‌بندی می‌شود [۱], [۶]. در این مقاله تمرکز ما بر روی زیر گروه "کنترل خودفرمان بینایی برمبنای تصویر" می‌باشد؛ که در آن تصاویر دو بعدی مستقیماً برای تخمین جابجایی مطلوب ربات استفاده می‌شوند [۷] و بدلیل پیاده‌سازی ساده‌تر، کارآمدتر می‌باشد [۱]. مبنای عملکرد این روش، کاهش خطای فاصله‌ای تصویری بین مجموعه‌ای از ویژگی‌های تصویری فعلی و مطلوب می‌باشد [۵]. اما معمولاً از الگوریتم‌هایی استفاده می‌شود که به عواملی همچون پارامترهای ذاتی دوربین[4]، پارامترهای بیرونی دوربین[5] و پارامترهای فیزیکی ربات وابسته هستند. دقت کالیبراسیون این پارامترها بصورت قابل توجهی بر خطای کنترل تأثیرگذار خواهد بود. همچنین کالیبراسیون کاری زمانبر و خسته‌کننده می‌باشد؛ پس استفاده از سیگنال‌های تصویری تنظیم نشده[6] مطلوب‌تر خواهد بود [۱].

با تقسیم‌بندیِ کنترل خودفرمان بینایی از دیدگاه دیگر، به دو گروه "سیستم‌های حرکت و بینایی دینامیک[7]" و "سیستم‌های خودفرمان مستقیم[8]" می‌رسیم. در گروه اول، سیستم بینایی، با انجام پردازش لازم بر روی تصاویر ورودی را برای کنترلگر ربات فراهم می‌کند. بسیاری از سیستم‌های موجود از این دسته می‌باشند. تمرکز ما در این مقاله هم بر روی این گروه با نصب یک دوربین ثابت می‌باشد. اما در گروه دوم، کنترلگر ربات حذف شده وکنترلگر بصری به‌طور مستقیم، ورودی را برای مفاصل ربات محاسبه می‌کند که بدلیل پیچیده شدن عملیات کنترلگر و سرعت کمتر نسبت به دسته اول، کاربرد کمتری دارد [۷], [۸].

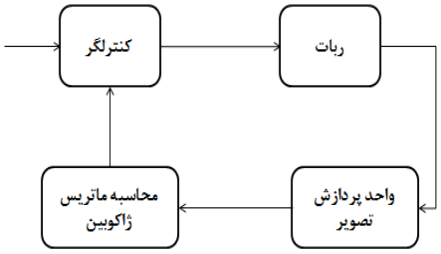
با مطالعه برروی کارهای انجام شده، روش‌های متنوعی در حوزه "سیستم‌های حرکت و بینایی دنیامیک" ارائه شده؛ که می‌توان آنها را به چهار گروه کلی تقسیم کرد:

۱) استفاده از مدل ربات (محیط)
۲) طراحی کنترلگر بصورت مستقیم
۳) استفاده از ماتریس ژاکوبین
۴) روش ترکیبی

در گروه اول، کنترل بر اساس مدل ربات انجام می‌شود و از جمله کارهای انجام شده، می‌توان به کنترل پیش‌بینی- مدل[9] اشاره کرد؛ که در [۹] بیان شده است. در بسیاری از موارد، بدست آوردن مدل از ربات، کاری دشوار و زمانبر خواهد بود و روش‌های این گروه استفاده کمی خواهند داشت.

در گروه دوم هدف این است که طراحی کنترلگر ربات بصورت مستقیم و با استفاده از ورودی‌ها و خروجی‌های متناظر، صورت گیرد. از جمله پژوهش‌های انجام شده در این گروه می‌توان به [۱۰]–[۱۲] اشاره کرد؛ که در آن‌ها به ترتیب از شبکه فازی-عصبی، سیستم فازی و یادگیری تقویتی[10] در بلوک کنترلگر استفاده شده است.

در مورد گروه سوم می‌توان گفت که در این گروه، فرمان کنترلی اعمالی به ربات، با توجه به قانون کنترل و با استفاده از ماتریس ژاکوبین تولید می‌شود (شکل ۲).



شکل ۲: دیاگرام "استفاده از ماتریس ژاکوبین"

[1] Visual Servoing
[2] Fixed Camera
[3] Eye-in-hand
[4] Intrinsic Parameter of Camera
[5] Extrinsic Parameter of Camera
[6] Uncalibrated Visual Signal
[7] Dynamic Look-and-Move Systems
[8] Direct Visual Servo Systems
[9] Model-prediction Control
[10] Reinforcement Learning

ماتریس ژاکوبین ماتریسی است که مؤلفه‌های آن نشان دهنده میزان تغییرات ویژگی‌ها به تغییر در مقادیر متغیرهای مفصل ربات، می‌باشد [۶], [۷], [۱۳] و به روش‌های متفاوتی محاسبه می‌شود. ساده‌ترین راه در این گروه این است که ماتریس ژاکوبین را با استفاده از روابط، بدست آورده و در هر تکرار از قوانین کنترلی، متغیرهای z و f در ماتریس ژاکوبین، که به ترتیب بیان کننده عمق فعلی دوربین و مقدار ویژگی‌های فعلی می‌باشند، ثابت و تقریبی از این مقادیر در محل مطلوب دوربین درنظرگرفته می‌شوند [۱۴]. این روش، ساده می‌باشد، سرعت همگرایی پایین و خطای بالایی دارد. در [۱۵] از فیلتر کالمن-بوسی[1] برای تخمین ماتریس ژاکوبین استفاده شده است. ضعف اساسی این کار در گسسته سازیِ انجام شده، می‌باشد.

در مقاله [۱۶] از دو شبکه عصبی یکی برای استخراج ویژگی و دیگری برای تخمین ماتریس ژاکوبین استفاده شده است؛ اما توضیحی در مورد قانون کنترلی استفاده شده و خروجی‌های هر کدام از شبکه‌ها بیان نشده است. در مقاله [۱۷] ورودی‌های شبکه عصبی تقریب زننده معکوس ژاکوبین، تنها وضعیت مجری نهایی است؛ که در عمل دیده می‌شود که با این وضعیت، شبکه عصبی نمی‌تواند تقریب مناسبی را ارائه دهد. همچنین در مقاله [۱۸] با استفاده از بحث پایداری لیاپانف، روابطی برای تازه سازی ماتریس ژاکوبین محاسبه شده است. شایان ذکر است مقالات [۱۶]-[۱۸] در یک محیط ساده شبیه سازی شده اند و در آنها مواردی چون نحوه پیدا کردن مفاصل در تصاویر، حذف نویز، لحاظ کردن تأخیر و دینامیکهای موتورهای محرکها لحاظ نشده است.

در مقاله [۱۳]، ماتریس ژاکوبین با استفاده از پروسه یادگیری و با طی مراحل زیر بدست می‌آید:

۱) بردن مجری‌نهایی به محل مطلوب (xd)
۲) فعال‌شدن حسگرها و بدست آوردن ویژگی‌های محل مطلوب (f_d)
۳) در مرحله آخر برای بدست آوردن ماتریس ژاکوبین، مجری‌نهایی در k موقعیت مکانی پشت‌سرهمِ از پیش تعریف شده اطراف محل مطلوب قرار می‌گیرد (این محل‌ها فاصله‌های یکسانی دارند و فقط با تغییر در r اُمین درجه آزادی[2] رخ می‌دهد) و در هر محل، بردار $f_s(r_i)$ (بردار ویژگی‌ها در محل) مقداردهی می‌شود.

ضرایب ستون مربوطه از ماتریس ژاکوبین (با توجه به اینکه کدام درجه آزادی یا متغیر مفصل ربات تغییر کرده است) را ثابت درنظرمی‌گیرد و در هر محل آموزش (r_i)، تمام ضرایب ستون مربوطه محاسبه می‌شوند. درنهایت برای بدست آوردن ضرایب ماتریس ژاکوبین از روش تخمین خطی استفاده می‌شود. توجه شود در مقاله مذکور از ساختار خطی برای بدست آوردن ماتریس ژاکوبین استفاده شده است؛ درحالیکه طبیعت سیستم، غیرخطی می‌باشد و لذا در عمل مشاهده می‌شود که دقت مطلوب به دست نمی‌آید.

در مورد گروه چهارم، که مقالات آن از ترکیب کنترل خودفرمان سنتی و روشی هوشمند استفاده می‌کنند، می‌توان به [۳] اشاره کرد؛ اما از مشکلات روش ارائه شده در این مقاله، می‌توان به محاسبات زیاد و گسسته‌سازی انجام شده، اشاره کرد.

در این مقاله هدف، طراحی یک سیستم برای کنترل مجری‌نهایی یک بازوی ربات واقعی مدل *RoboTEK II ST240* [۱۹] جهت رسیدن از یک نقطه شروع به نقطه هدف با استفاده از تنها یک دوربین ساده (وبکم) نصب شده در محل می‌باشد. طبق مطالب بیان شده، بدلیل دردسترس نبودن مدل ربات و گریز از مشکلات پیشرو برای تقریب آن، همچنین گریز از محاسبات و ساختار پیچیده، در اینجا از گروه سوم (استفاده از ماتریس ژاکوبین) استفاده خواهیم کرد. با توجه به ساختار کنترلگر انتخابی (که از معکوس ماتریس ژاکوبین استفاده می‌کند) و ماهیت غیرخطی ماتریس ژاکوبین، از شبکه عصبی برای تقریب معکوس ماتریس ژاکوبین استفاده می‌شود.

از جمله مهمترین چالش‌هایی که در عمل با آن مواجه هستیم، می‌توان به اختصار موارد زیر را بیان کرد؛ که در این مقاله راهکارهای مناسب برای هرکدام از آن‌ها ارائه خواهد شد:

۱) تعیین ویژگی‌های مناسب قابل استخراج از تصاویر
۲) تعیین ساختار کنترلگر
۳) نحوه محاسبه ماتریس ژاکوبین و ارائه ساختار هوشمند برای تقریب آن
۴) تشخیص محل مفاصل بازوی ربات (حالت سیستم) با استفاده از تصاویر

ساختار مقاله به این شرح می‌باشد: در بخش دوم مفاهیم پایه بیان می‌شود. روش ارائه شده در بخش سوم می‌آید. بخش چهارم حاوی نتایج حاصل از آزمایشات انجام شده بر روی یک بازوی ربات واقعی است و بخش پنجم بیان کننده نتیجه‌گیری می‌باشد.

۲- مفاهیم پایه

همانگونه که قبلاً هم بیان شد، ماتریس ژاکوبین ماتریسی است که مؤلفه‌های آن نشان دهنده میزان تغییرات ویژگی‌ها به تغییر در مقادیر متغیرهای مفصل ربات می‌باشد؛ پس می‌توان آن را بصورت زیر نمایش داد [۶], [۷], [۱۳]:

$$J(q)=\left[\frac{\partial f}{\partial q}\right]=\begin{bmatrix}\frac{\partial f_1(q)}{\partial q_1} & \cdots & \frac{\partial f_1(q)}{\partial q_m}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_k(q)}{\partial q_1} & \cdots & \frac{\partial f_k(q)}{\partial q_m}\end{bmatrix} \qquad (۱)$$

[1] Kalman-Bucy filter
[2] Degree of Freedom (DOF)

در رابطه فوق، J ماتریس ژاکوبین، m نشان دهنده درجات آزادی ربات، q نشان دهنده متغیرهای مفصل ربات، k تعداد ویژگی‌های تصویری و f برداری از ویژگی‌های تصویری می‌باشد.

۲-۱ روابط ماتریس ژاکوبین

با توجه به آنچه در [۲]، [۳]، [۷]، [۲۰] بیان شده و با فرض اینکه سرعت زاویه‌ای (${}^{c}\Omega_e$) و سرعت خطی (${}^{c}T_e$) مجری‌نهایی با توجه به قاب دوربین به ترتیب ${}^{c}\Omega_e=[w_x,w_y,w_z]^T$ و ${}^{c}T_e=[v_x,v_y,v_z]^T$ باشند (x، y و z محورهای مختصات و w_x و v_x به ترتیب سرعت زاویه‌ای و سرعت خطی مجری‌نهایی در راستای محور x می‌باشد.) و نقطه P نقطه‌ای باشد که به مجری‌نهایی متصل است؛ سرعت نقطه P با توجه به قاب دوربین طبق رابطه (۲) می‌باشد:

$$ {}^{c}\dot{P}={}^{c}\Omega_e\times{}^{c}P+{}^{c}T \qquad (۲)$$

که می‌توان بصورت سیستم سه معادله‌ای زیر نوشت:

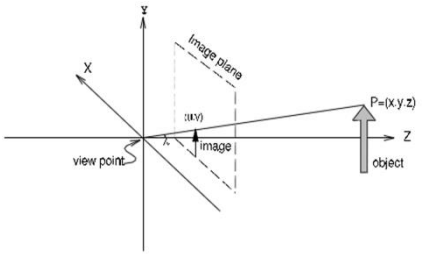
$$\dot{x} = z\,w_y - y\,w_z + v_x \qquad (۳)$$

$$\dot{y} = x\,w_z - z\,w_x + v_y \qquad (۴)$$

$$\dot{z} = y\,w_x - x\,w_y + v_z \qquad (۵)$$

از آنجا که u و v مختصه‌های تصویر نقطه P در تصویر دوربین می‌باشند، با توجه به شکل ۳ و روابط در دو مثلث مشابه، روابط زیر را داریم:

$$x = \frac{uz}{\lambda}, y = \frac{vz}{\lambda} \qquad (۶)$$



شکل۳: دستگاه مختصات دوربین [۳]

با جاگذاری روابط (۶) در روابط (۳) تا (۵) داریم:

$$\dot{x} = z\,w_y - \frac{vz}{\lambda}w_z + v_x \qquad (۷)$$

$$\dot{y} = \frac{uz}{\lambda}w_z - z\,w_x + v_y \qquad (۸)$$

$$\dot{z} = \frac{z}{\lambda}(v\,w_x - u\,w_y) - v_z \qquad (۹)$$

با استفاده از قاعده مشتق‌گیری از رابطه کسری و روابط (۶) خواهیم داشت:

$$\dot{u} = \lambda\frac{z\dot{x} - x\dot{z}}{z^2} \qquad (۱۰)$$

$$\dot{v} = \lambda\frac{z\dot{y} - y\dot{z}}{z^2} \qquad (۱۱)$$

با جاگذاری روابط (۱۰) و (۱۱) در روابط فوق خواهیم داشت:

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix}=\begin{bmatrix}\frac{\lambda}{z} & 0 & \frac{-u}{z} & \frac{-uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v\\ 0 & \frac{\lambda}{z} & \frac{-v}{z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ v_z\\ w_x\\ w_y\\ w_z\end{bmatrix} \qquad (۱۲)$$

که ماتریس ۶×۲ فوق ماتریس ژاکوبین نامیده می‌شود و رابطه (۱۲) را می‌توان بصورت مختصر در رابطه (۱۳) نشان داد:

$$\dot{f} = J\dot{q} \qquad (۱۳)$$

در این رابطه، f نشان دهنده تغییرات ویژگی‌های تصویری می‌باشد. درنهایت با استفاده از رابطه (۱۳)، می‌توان تخمینی از سرعت ربات ($\dot{q}=[v_x,v_y,v_z,w_x,w_y,w_z]^T$) بدست آورد؛ تا ویژگی‌ها در صفحه تصویر فعلی به ویژگی‌های مطلوب برسند. به بیان دیگر با این رابطه، حرکت لازم ربات بدست می‌آید؛ تا ربات از محل شروع به محل مطلوب (هدف) برسد. ویژگی‌های مطلوب با روش آموزش با نشان دادن[1]، به ربات نشان داده می‌شوند و این ویژگی‌ها ذخیره می‌شوند.

۲-۲ فرمان کنترل

با استفاده از روابط و توضیحات ارائه شده در [۲]، [۳]، [۵]، [۶]، [۱۳]، رابطه بین تغییرات ویژگی‌های تصویری ($\dot{f}$) و تغییرات مختصات مفاصل ربات ($\dot{q}$) را می‌توان با رابطه (۱۳) بیان کرد. اگر محل تصویری نقاط ویژگی به عنوان معیار سنجش استفاده شود، آنگاه تابع خطا بصورت رابطه (۱۴) تعریف می‌شود:

$$e(f) = f^c - f^* \qquad (۱۴)$$

در رابطه فوق f^c نشان دهنده بردار ویژگی در موقعیت فعلی و f^* نشان دهنده بردار ویژگی مطلوب می‌باشد. هدف کنترل، صفر شدن تابع خطای فوق می‌باشد و با این هدف به رابطه زیر خواهیم رسید:

$$u = \dot{q} = K\,J^{\#}e(f) \qquad (۱۵)$$

که در آن u نشان دهنده فرمان کنترلی، $J^{\#}$ نشان دهنده معکوس کلی ماتریس ژاکوبین و K ماتریسی با مقادیر ثابت است که ماتریس بهره نامیده می‌شود. بدست آوردن معکوس کلی ماتریس ژاکوبین براساس بعد فضای ویژگی (p) و بعد فضای کاری (m)، بصورت زیر انجام می‌شود:

[1] Teach by Showing

$$J^{\#}=\begin{cases} J^T.(J.J^T)^{-1}, m>p \\ (J^T.J)^{-1}.J^T, m<p \\ J^{-1}, m=p \end{cases} \qquad (۱۶)$$

چالش اصلی موجود، تعیین ویژگی‌های مرتبط و بدست آوردن ماتریس ژاکوبین می‌باشد؛ تا با استفاده از رابطه (۱۵) فرمان کنترلی مناسب تولید شود.

۳- روش ارائه شده

طبق رابطه (۱)، هریک از المانهای ماتریس ژاکوبین بیان کننده میزان تغییرات ویژگی‌ها به تغییر در مقادیر متغیرهای مفصل ربات می‌باشد. در این مقاله، ابتدا داده‌های مناسب جمع‌آوری می‌شود. سپس با توجه به ماهیت غیرخطی ماتریس ژاکوبین، از شبکه‌های عصبی برای تقریب معکوس ماترس ژاکوبین استفاده می‌شود. از این مدل تقریب زده شده توسط شبکه عصبی، برای بدست آوردن فرمان کنترلی طبق رابطه (۱۵) استفاده می‌گردد. داده‌ها برای تقریب معکوس ماتریس ژاکوبین، باید نشان دهنده میزان تغییرات ویژگی‌ها به تغییر در مقادیر متغیرهای مفصل ربات باشند. پس ابتدا بازوی ربات در l موقعیت ابتدایی متمایز قرار داده می‌شود، در بعدهای مختلف حرکت داده می‌شود و داده‌های لازم جمع‌آوری می‌شود. به بیانی دیگر هر بار، با تغییر یک درجه‌ای در مقادیر یکی از متغیرهای مفاصل، میزان تغییرات ویژگی‌ها (که المانهای یک ستون مشخص در ماتریس ژاکوبین محسوب می‌شوند) جمع‌آوری می‌شود. پس از آن، به ازای هرکدام از درجات آزادی بازوی ربات، یک شبکه عصبی دو لایه با ساختار پیشرو طراحی می‌شود. ورودی‌های هر کدام از شبکه‌ها (بردار A در رابطه (۱۷))، مقادیر ویژگی‌ها (F) و حالت سیستم (S) می‌باشند و خروجی‌ها بیان کننده میزان تغییر در مقادیر ویژگی‌ها برحسب میزان تغییر در متغیر مفصل موردنظر می‌باشند. درنهایت شبکه‌ها با داده‌های جمع‌آوری شده و بصورت برون‌خط، آموزش داده می‌شوند. ماتریس B در رابطه (۱۸)، معکوس ماتریس ژاکوبین تقریب زده شده، می‌باشد؛ که ابعاد آن n (تعداد درجات آزادی بازوی ربات) در m (تعداد ویژگی‌های انتخاب شده) می‌باشد و هر سطر آن، توسط یکی از شبکه‌های عصبی تقریب زده می‌شود.

$$A=[S,F] \qquad (۱۷)$$

$$B=\begin{bmatrix} \frac{\Delta a}{\Delta f_1} & \cdots & \frac{\Delta a}{\Delta f_m} \\ \vdots & \ddots & \vdots \\ \frac{\Delta z}{\Delta f_1} & \cdots & \frac{\Delta z}{\Delta f_m} \end{bmatrix} \qquad (۱۸)$$

انتخاب ویژگی‌ها باید با دقت صورت گیرد؛ تا حرکت شی را بصورت یکتا نشان دهد، مقدار آن‌ها در محل مطلوب مشخص باشد [۷] و باعث ایجاد ابهام نشود. در اولین شبکه‌های طراحی شده، از فاصله اقلیدسی مجری‌نهایی با محل مطلوب، به عنوان ویژگی استفاده شد،؛ اما به جواب قابل قبولی نرسیدیم. بنابراین فاصله مجری‌نهایی در بعد x و y با هدف به عنوان ویژگی‌ها در این مسئله، انتخاب شدند. ورودی دیگر شبکه عصبی، باید نشان دهنده حالت سیستم باشد؛ که با موقعیت سه مفصل بازوی ربات بیان می‌شود. اما بخاطر محل قرارگیری دوربین (که تقریباً در راستای خط تقارن پایه ربات قرار گرفته)، موقعیت مفصل پایه ربات بصورت ضمنی تأثیرگذار خواهد بود و از آنجا که دیگر ورودی‌های شبکه، فاصله مجری‌نهایی تا هدف (که همان ویژگی انتخاب شده می‌باشد) را نشان می‌دهد؛ پس موقعیت مجری‌نهایی نیز در شبکه وارد می‌شود. پس برای بیان منحصربه‌فرد حالت سیستم، لازم است موقعیت مفصل شانه ربات، به عنوان ورودی به شبکه داده شود. با توجه به توضیحات ذکر شده، برای تأمین ورودی‌های شبکه‌ها، باید محل مفصل شانه و مجری‌نهایی با استفاده از تصاویر دوربین، مشخص شود. برای این منظور روش‌های متفاوتی بررسی شد و یکی از آن‌ها ایجاد الگوی متمایزی همچون سه مثلث با آرایش خاص، برای یافتن محل این دو مفصل می‌باشد. اما از آنجا که در این کار، تنها از یک دوربین استفاده می‌شود؛ در برخی از نقاط همچون گوشه‌های فضای کاری، آرایش خاص مورد نظر در تصویر دیده نمی‌شود و تشخیص الگو امکان‌پذیر نخواهد بود. روش دیگری که بررسی شد، استفاده از نشانگری مانند مستطیل با رنگی خاص و یافتن آن در تصاویر گرفته شده، بود. در اینجا با توجه به امکانات در دسترس آزمایشگاه، از خمیرهای رنگی برای تولید مستطیل رنگی بعنوان نشانگر استفاده شد. این نشانگرها روی مفاصل قرار داده شده و برای مشخص کردن محل مفاصل، کافی است محل این نشانگرها مشخص شود. اما بدلیل اینکه محیط کار شامل رنگ‌های متنوعی می‌باشد؛ یافتن رنگ موردنظر دشوار خواهد بود. برای مشخص کردن محل مفصل شانه، می‌توان فضای جستجو را محدودتر کرد و از الگوریتم رشد ناحیه[1] استفاده کرد. بدین ترتیب که در شروع کار فقط یک نقطه از خمیر مفصل شانه، لازم است و با اعمال شرط رنگی سخت‌گیرانه‌ای، این نقطه درست تشخیص داده می‌شود. سپس با استفاده از الگوریتم رشد ناحیه، کل خمیر، بدست آمده و نقطه مرکز آن به عنوان محل مفصل درنظرگرفته خواهد شد. پس از تولید فرمان کنترلی و حرکت بازو، مجدداً باید این خمیر مشخص شود. این بار در پنجره‌ای با ابعاد ۱۰× ۱۰ به دنبال نقطه‌ای هستیم؛ که به رنگ مرکز خمیر مربوط به مفصل شانه، شبیه باشد (یعنی اختلافی حداکثر ۴۰ واحدی در سه ماتریس رنگی آن‌ها وجود داشته باشد). سپس با استفاده از الگوریتم رشد ناحیه، کل خمیر درنظر گرفته شده برای مفصل شانه پیدا خواهد شد. لازم به ذکر است، الگوریتم رشد ناحیه بدین ترتیب عمل می‌کند که همسایگی

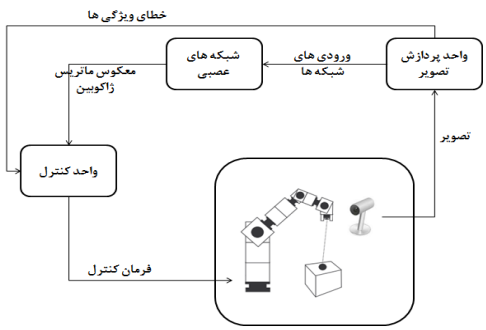
[1] Region Growing Algorithm

۸ تایی اطراف نقطه مرجع را بررسی می‌کند و هرکدام از همسایه‌ها که از نظر رنگی به اندازه کافی به نقطه مرجع شبیه باشد (میزان شباهت با استفاده از یک ضریب بیان می‌شود)، به نقاط اضافه می‌شود و اینکار ادامه می‌یابد تا ناحیه پیوسته رنگی‌ای در اطراف نقطه مرجع بدست آید.

مرحله بعد، محاسبه خروجی قانون کنترل می‌باشد. ابتدا با پردازش تصویرِ حاصل از دوربین، داده‌های لازم (مقادیر ویژگی‌ها و حالت سیستم (بازوی ربات) در محل فعلی) بدست می‌آیند؛ که هر دو ورودی‌های شبکه‌ها هستند. با استفاده از مقادیر ویژگی‌ها در محل مطلوب، خطای این ویژگی‌ها در محل فعلی محاسبه می‌شود. سپس با اعمال ورودی‌ها به شبکه‌ها، خروجی‌های شبکه‌ها (همان المان‌های معکوس ماتریس ژاکوبین در محل فعلی) محاسبه می‌شوند. درنهایت با استفاده از رابطه (۱۹) فرمان کنترلی مناسب یعنی میزان تغییر در متغیرهای مفاصل بازوی ربات تولید و به ربات اعمال می‌شود.

$$u_i = \Delta q_i = k_i \times NN_i(A) \times e(F), i = 1..n \qquad (۱۹)$$

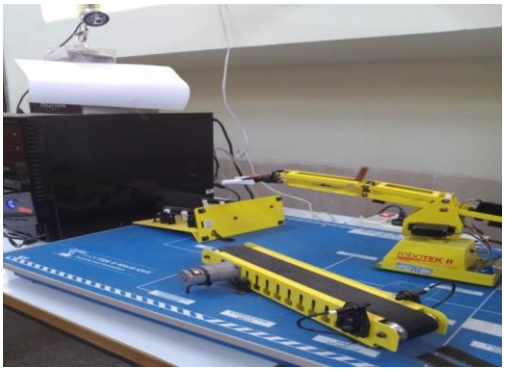
در رابطه فوق i نشان دهنده شماره متغیر مفصل بازوی ربات و n نشان دهنده تعداد درجات آزادی (موتورهای مفاصل) ربات می‌باشد. طبق رابطه (۱۹) با ضرب خروجی‌های هر شبکه، که نشان دهنده سطری از معکوس ماتریس ژاکوبین در محل فعلی است که مربوط به متغیر مفصل موردنظر می‌باشد، در سطر مربوطه از ماتریس با مقادیر ثابت K و خطای ویژگی‌های درنظرگرفته شده $e(F)$، فرمان کنترلی مناسب حاصل می‌شود؛ که این فرمان نشان دهنده میزان تغییر در مقدار متغیر مفصل موردنظر بازوی ربات(Δq_i) می‌باشد. قابل ذکر است که ماتریس K، ماتریس قطری با مقادیر ثابت بین ۰ تا ۱ می‌باشد. نتیجه اعمال این فرمان به بازوی ربات، حرکت آن در جهت رسیدن به هدف می‌باشد و همین عملیات در موقعیت جدید انجام می‌شود؛ تا مجری‌نهایی به هدف از پیش تعیین شده برسد. کلیات روش ارائه شده را می‌توان در شکل زیر (شکل۴) مشاهده کرد.



شکل۴: کلیات روش ارائه شده

۴- آزمایشات

در این مقاله از بازوی ربات واقعی *RoboTEK II ST240* [۱۹] استفاده شده است؛ که ربات مذکور با سه درجه آزادی دورانی از گروه ربات‌های آرنجی می‌باشد و نرم‌افزار لازم برای کنترل آن نوشته شد. به منظور گرفتن تصاویر، که ورودی‌های واحد پردازش تصویر می‌باشند، از یک دوربین (وبکم ساده با قدرت تفکیک[1] ۹۶ پیکسل در هر اینچ) که بصورت ثابت نصب شده، استفاده شده است. نمای کلی سیستم در شکل ۵ نشان داده شده است و فیلم‌هایی از آزمایشات انجام شده در سایت آزمایشگاه هوش ماشین و رباتیک دانشگاه یزد [۲۱] قرار دارد.



شکل۵: نمایی کلی از سیستم

به منظور بدست آوردن المان‌های معکوس ماتریس ژاکوبین از سه شبکه عصبی، هرکدام برای یکی از متغیرهای مفصل بازوی ربات، استفاده شده است. هر شبکه، چهار ورودی (فاصله مجری‌نهایی در بعد x و y با هدف و موقعیت مفصل شانه بازوی ربات) و دو خروجی (سطر متناظر با مفصل مربوطه در معکوس ماتریس ژاکوبین $\frac{\Delta a}{\Delta f_1}$ و $\frac{\Delta a}{\Delta f_2}$) خواهد داشت. برای تأمین ورودی‌های شبکه‌ها، باید محل مفصل شانه و مجری‌نهایی بازوی ربات با استفاده از تصاویر وبکم، مشخص شود. ابتدایی‌ترین چالش در این راستا، تار بودن تصویر دوربین در ابتدای فراخوانی و خواندن تصویر آن بود و به این دلیل رخ می‌داد که باید مدتی صبر می‌کردیم؛ تا ارتباط بین کامپیوتر و دوربین پایدار شود. پس با اضافه کردن تأخیر ۱۰ ثانیه‌ای در ابتدای عملیات، این چالش حل شد. چالش دیگر تأثیر نور موجود در محیط، بر میزان روشنایی تصاویر حاصل بود؛ که در ابتدا با استفاده از روش متعادل کردن هیستوگرام تصویر، نتیجه قابل قبول حاصل نشد و با تقسیم کردن تمام پیکسل‌ها بر ماکزیمم مقادیر پیکسل‌ها به نتیجه قابل‌قبولی رسیدیم و تاثیر نور کمرنگ‌تر شد.

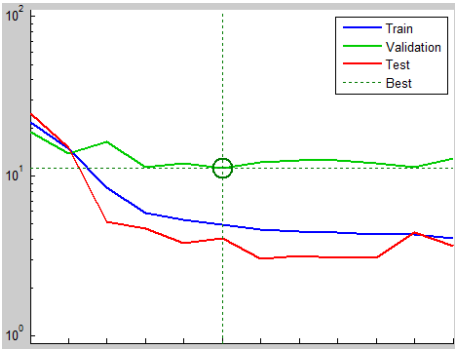
برای جمع‌آوری داده‌های آموزشی شبکه‌ها، بازوی ربات در ۷ موقعیت ابتدایی متمایز قرار داده شد و هر بار با تغییر ۵ درجه‌ای در مقادیر یکی از سه مفصل، ورودی‌ها و خروجی‌های شبکه متناظر ذخیره شد. قابل ذکر است که بدلیل پایین بودن قدرت تفکیک دوربین استفاده شده، گاهی تغییر کمتر از ۵ درجه در مفصل‌ها، در تصاویر حاصل، قابل مشاهده نبود. با ۲۸۷ داده بدست آمده، شبکه‌های عصبی آموزش داده شد. در جدول ۱ مجذور میانگین مربعات خطای[2] آموزش شبکه‌های مذکور آمده است.

[1] Resolution
[2] Root Mean Square Error (RMSE)

جدول ۱: مجذور میانگین مربعات خطای آموزش هر شبکه

شبکه آموزش داده شده	مجذور میانگین مربعات خطای آموزش
شبکه اول (مربوط به مفصل پایه ربات)	۲/۴
شبکه دوم (مربوط به مفصل شانه ربات)	۲/۱
شبکه سوم (مربوط به مفصل آرنج ربات)	۲/۲

شکل ۶ تغییرات تابع هزینه (میانگین مربعات خطا) در حین آموزش را برای شبکه اول (مربوط به مفصل پایه ربات) نشان می‌دهد. این شبکه عصبی در طی ۱۵ تکرار توانسته است خروجی را با خطای قابل قبولی تقریب بزند. همچنانکه در شکل دیده می‌شود، خطا برای داده تست نیز بسیار پایین (*RMSE=* ۲/۴) می‌باشد که حاکی از قابلیت تعمیم شبکه عصبی تنظیم شده می‌باشد.



شکل۶: تغییرات تابع هزینه در حین آموزش برای شبکه اول

در ادامه از شبکه‌های عصبی آموزش دیده، برای بدست آوردن مقدار فرمان موتورها طبق رابطه (۱۹) استفاده می‌شود.

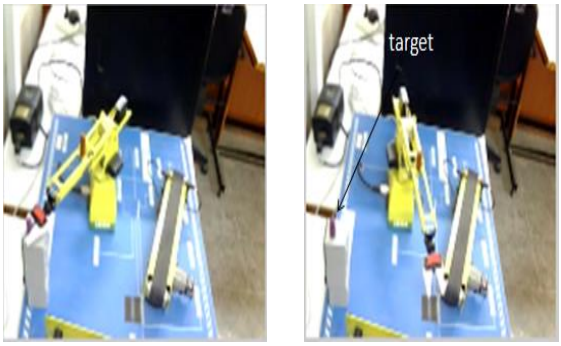
برای تعیین مقادیر ماتریس K در رابطه (۱۹) باید توجه کرد که نباید خیلی کوچک درنظر گرفته شود؛ چون سرعت رسیدن به هدف (صفر شدن بردار خطا) کم خواهد شد. همچنین نباید خیلی هم بزرگ انتخاب شود؛ چون بازه حرکت ربات در هر چرخه کنترلی زیاد خواهد شد و اگر خطای اندکی در فرمان اعمالی وجود داشته باشد، تبدیل به جابجایی بزرگی می‌شود. پس امکان دارد تعداد گام‌ها در رسیدن به هدف زیاد شود و یا ربات از هدف عبور کند. با انجام چند آزمایش و مقایسه نتایج، ماتریس K بصورت زیر درنظر گرفته شد:

$$K = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix} \qquad (۲۰)$$

۴-۱ بیان جزئی‌تر آزمایشات

در سیستم رباتی که در اختیار است، مقدار مجازی که می‌توان به عنوان فرمان به محرک هر مفصل ربات وارد کرد؛ عددی صحیح بین ۱۲۷- تا ۱۲۷ می‌باشد. همچنین آزمایشات نشان می‌دهد هر یک واحد فرمان کنترلی باعث تغییر ۵ پیکسلی در تصویر می‌شود. در نتیجه حداقل خطای مورد انتظار ۵ پیکسل می‌تواند باشد و کمتر از آن قابل وصول نمی‌باشد. به عنوان مثال اگر خطا ۱- پیکسل باشد، با یک واحد تغییر در مفصل، تغییر ۵ پیکسلی در تصویر ایجاد و خطا ۴ پیکسل می‌شود. البته این در بهترین وضعیت و بدون لحاظ کردن نویز تصویر و نایقینی‌های قسمت کنترل است. توجه شود که در اینجا از یک دوربین ساده (وبکم) استفاده شده و لذا قدرت تفکیک هم بالا نیست. ضمن آنکه استفاده از تنها یک دوربین (بجای استریو) می‌تواند خود باعث قرار گرفتن دو نقطه متفاوت روی هم در تصویر گرفته شده، بشود.

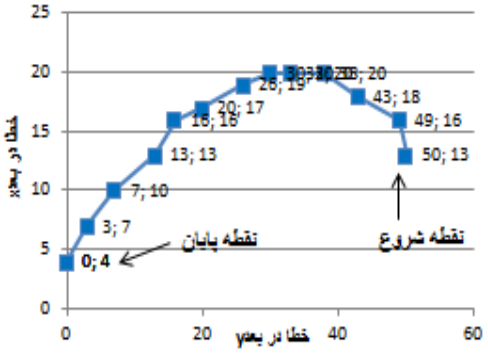
مقادیر مفاصل ربات در اولین آزمایش و در ابتدای حرکت ۵۰،۵۰ و ۶۰ می‌باشد (یا مختصات مجری‌نهایی در قاب دوربین (۸۰،۹۳) می‌باشد) و هدف در نقطه با مختصات (۸۰،۳۰) قرار دارد. در شکل ۷ (الف)، محل قرارگیری بازوی ربات در ابتدای عملیات و هدف تعیین شده، مشخص است و با گذشت ۳۰/۳ ثانیه، مجری‌نهایی با دقت خوبی به نقطه نهایی رسیده است (شکل ۷ (ب)). در این مثال درنهایت فاصله مجری‌نهایی تا هدف در بعد x، ۴ پیکسل و در بعد y، صفر پیکسل می‌باشد. قابل ذکر است که زمان اعلام شده با احتساب ۱۰ ثانیه برای پایداری ارتباط دوربین و کامپیوتر می‌باشد و به ازای هر فرمان ارسالی به ربات، لازم است ۲ ثانیه تأخیر اعمال شود؛ تا ربات فرمان را بصورت کامل انجام دهد و به موقعیت جدید برسد. همچنین اندازه ماتریس تصویر ۱۲۰×۱۶۰ پیکسل می‌باشد.



(الف) (ب)

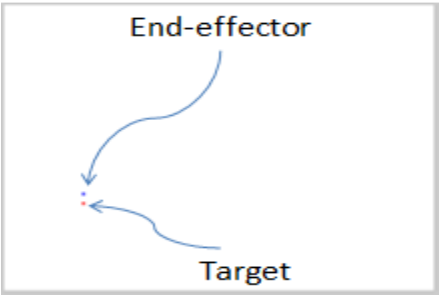
شکل۷ الف: موقعیت ابتدایی شکل۷ ب: موقعیت نهایی

همانطور که در شکل ۸ مشخص است، در ابتدای حرکت، خطای بین مختصات مجری‌نهایی با هدف در بعد x و y به ترتیب ۱۳ و ۵۰ می‌باشد. با اعمال اولین فرمان کنترلی به بازوی ربات و حرکت آن به سمت هدف (در اولین گام از پروسه کنترل بازو)، خطای مجری‌نهایی با هدف (۴۹،۱۶) می‌شود. حرکت ربات با توجه به فرمان کنترلی اعمالی ادامه می‌یابد و خطای بین مجری‌نهایی و هدف، به کمتر از ۵ پیکسل کاهش می‌یابد؛ تا مجری‌نهایی به نقطه (۳۰،۸۴) می‌رسد.



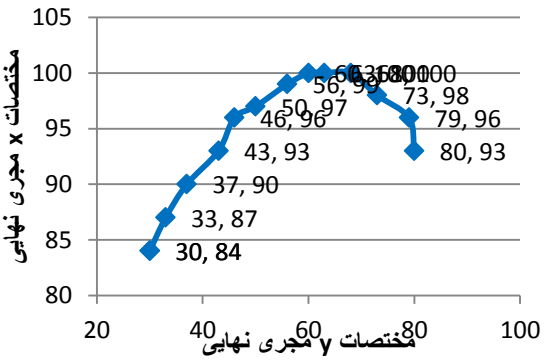
شکل۸: خطای ویژگی‌های تصویری برحسب پیکسل

شکل ۹ تصویری با ابعاد ۱۲۰×۱۶۰ پیکسل (هم سایز با تصویر دوربین) به رنگ سفید را نشان می‌دهد؛ که در آن، پیکسل مربوط به نقطه هدف و پیکسل مربوط به نقطه مجری‌نهایی بترتیب با رنگ قرمز و سبز مشخص شده است. همچنانکه دیده می شود فاصله این دو پیکسل کمتر از ۵ واحد در هر بعد می باشد.



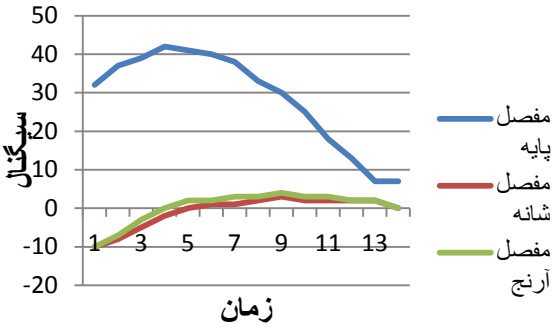
شکل۹: خطای محل مجری‌نهایی با هدف در موقعیت نهایی با دید پیکسلی

در شکل ۱۰ مسیر حرکت مجری‌نهایی از نقطه شروع (۹۳،۸۰) نشان داده شده است. با استفاده از روش ارائه شده، مجری‌نهایی به نقطه با مختصات (۸۴،۳۰) می‌رسد؛ که در فاصله ۴ و صفر پیکسلی به ترتیب در بعد x و y با هدف (با مختصات (۸۰،۳۰)) می‌باشد.



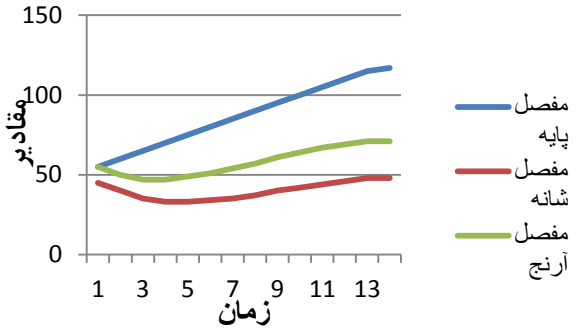
شکل۱۰: مسیر حرکت مجری نهایی ربات

شکل۱۱ در مورد سیگنال کنترلی اعمالی به هر یک از موتورهای بازوی ربات می‌باشد و طبق نمودار، در ابتدای حرکت که فاصله مجری‌نهایی با هدف زیاد است، تغییرات زیاد می‌باشد اما درنهایت سیگنال‌های کنترلی کوچک می‌شود تا تغییرات اندک در محل مجری‌نهایی رخ دهد و خطا کم شود.



شکل۱۱: سیگنال کنترلی

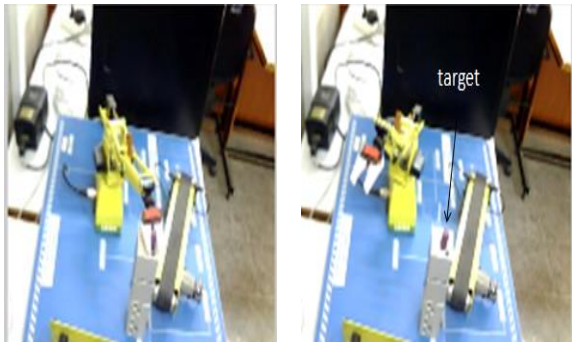
در شکل ۱۲ مقادیر متغیرهای مفصل ربات از نقطه ابتدایی تا هدف نشان داده شده است. همانگونه که مشخص می‌باشد، در ابتدای حرکت مقادیر مفاصل ۵۰، ۵۰ و ۶۰ می‌باشد و اعمال سیگنال کنترلی باعث تغییر در این مقادیر می‌شود؛ تا مجری‌نهایی از نقطه شروع به هدف برسد.



شکل ۱۲: مقادیر مفاصل ربات

در عمل و برای جلوگیری از نوسانات حول هدف، شرطی گذاشتیم که وقتی خطا به ۵ پیکسل رسید؛ ربات متوقف شود و فرمانی به آن اعمال نشود. لیکن از آنجا که در تصاویر گرفته شده پیکسل هدف و مجری نهایی دقیقاً منطبق نیستند (طبق شکل ۹)؛ خروجی ماتریس ژاکوبین و در نتیجه فرمان کنترلی صفر نخواهد شد (طبق شکل ۱۱). برای بهبود، تنها راه استفاده از رباتی است که دقت حرکت بالاتری داشته باشد بدین معنی که با یک واحد فرمان تغییرات کوچکتری را در بعد مربوطه داشته باشد.

تصاویر ابتدایی و انتهایی سیستم در آزمایش دیگری، به ترتیب در شکل ۱۳ (الف) و ۱۳ (ب) نشان داده شده است. در این آزمایش، مقادیر مفاصل ربات در ابتدای حرکت، ۸۰،۱۰۰ و ۱۰۰ می‌باشد؛ به بیان دیگر مختصات مجری‌نهایی در قاب دوربین و در شروع، (۵۴،۴۳) می‌باشد و هدف در نقطه با مختصات (۸۰،۸۰) قرار دارد. در این مثال با گذشت ۲۹/۸ ثانیه، فاصله مجری‌نهایی تا هدف در بعد x، ۳- پیکسل و در بعد y، ۴ پیکسل می‌شود.



(الف) (ب)

شکل۱۳ الف: موقعیت ابتدایی شکل۱۳ ب: موقعیت نهایی

۵- نتیجه‌گیری

در این مقاله راهکاری جهت کنترل ربات با تصویر، برای رسیدن از هر نقطه شروع دلخواه به هدف تعیین شده پیشنهاد شد. قانون کنترل درنظر گرفته شده برای این منظور، مبتنی بر ماتریس ژاکوبین می‌باشد و از معکوس ماتریس ژاکوبین برای محاسبه فرمان کنترلی استفاده می‌کند. از آنجا که اطلاعاتی از ربات و پارامترهای دوربین دردسترس نبود؛ با استفاده از شبکه عصبی و آموزش برون‌خط آن با داده‌های جمع‌آوری شده، معکوس ماتریس ژاکوبین تقریب زده شد. به ازای هرکدام از درجات آزادی ربات، یک شبکه عصبی درنظر گرفته شد. اولین انتخاب برای ویژگی‌ها (ورودی‌های دیگر شبکه‌های عصبی)، فاصله اقلیدسی مجری‌نهایی با هدف بود؛ اما نتایج قابل قبولی حاصل نشد. لذا در تلاش بعدی فاصله مجری‌نهایی در بعد x و y با هدف، و مختصات مفصل شانه در دو بعد x و y به عنوان ورودی درنظر گرفته شدند. نتایج نشان داد که شبکه عصبی با ورودی های مذکور میتواند تقریب خوبی از معکوس ماتریس ژاکوبین بزند. برای حل چالش مشخص کردن مختصات مفصل شانه، این ایده داده شد که از یک پنجره متحرک و الگوریتم رشد ناحیه استفاده شود. جهت بررسی عملکرد روش مذکور، از بازوی ربات واقعی مدل RoboTEK II ST240 [۱۹] استفاده شد. در راستای پیاده‌سازی، چالش‌های متعددی از جمله چگونگی پیدا کردن محل مفاصل با استفاده از تصویر دوربین، تار بودن تصویر دوربین در ابتدای فراخوانی، و تأثیر نور محیط وجود داشت؛ اما برای هر کدام راهکار مناسب ارائه شد. در نهایت آزمایشات نشان داد که روش ارائه شده میتواند مجری نهایی را از هر مکان در فضای مورد بررسی به نقطه دلخواه با دقت قابل قبول برساند. قابل ذکر است که ساده بودن تجهیزات استفاده شده، می‌تواند نشان دهنده کارآیی روش ارائه شده باشد. بدیهی است افزایش دقت موتورهای بازوهای ربات و کیفیت تصویر می تواند باعث بهبود کارآیی گردد.

مراجع

[1] H. Wang, M. Jiang, W. Chen, and Y. Liu, “Visual servoing of robots with uncalibrated robot and camera parameters”, *Mechatronics*, vol. 22, no. 6, pp. 661–668, 2012.

[2] S. Hutchinson and F. Chaumette, “Visual Servo Control”, *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 82–90, 2006.

[3] M. Mitic, M. Lazarevic, B. Babic, and Z. Miljkovic, “Neural network Reinforcement Learning for visual control of robot manipulators”, *Expert System with Application*, vol. 40, no. 5, pp. 1721–1736, 2013.

[4] H. Wang, Y. Liu, and W. Chen, “Visual tracking of robots in uncalibrated environments,” *Mechatronics*, vol. 22, no. 4, pp. 390–397, 2012.

[5] P. P. Kumar and L. Behera, “Visual servoing of redundant manipulator with Jacobian matrix estimation using self-organizing map”, *Robotics and Autonomous Systems*, vol. 58, no. 8, pp. 978–990, 2010.

[6] D. Kragic, H. I. Christensen, and Others, “Survey on Visual Servoing for Manipulation”, *Computational Vision and Active Perception Laboratory*, vol. 15, no. 1, 2002.

[7] S. Hutchinson, G. D. Hager, and P. Corke, “A Tutorial on Visual Servo Control”, *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 651–670, 1996.

[8] D. Kragic, H. I. Christensen, and Others, “Survey on Visual Servoing for Manipulation”, *Computational Vision and Active Perception Laboratory*, vol. 15, no. 1, 2002.

[9] G. Allibert and E. Courtial, “Predictive Control for Constrained Image-Based Visual Servoing”, *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, 2010.

[10] I. H. Suh and T. W. Kim, “Fuzzy membership function based neural networks with applications to the visual servoing of robot manipulators”, *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 3, pp. 203–220, 1994.

[11] I. Siradjuddin, L. Behera, T. M. McGinnity, and S. Coleman, “Image Based Visual Servoing of a 7 DOF robot manipulator using a distributed fuzzy proportional controller”, *IEEE International Conference on Fuzzy Systems (FUZZ)*, pp. 1–8, 2010.

[12] Y. Takahashi, M. Takeda, and M. Asada, “Continuous Valued Q-learning for Vision-Guided Behavior Acquisition”, *International Conference on Multisenso Fusion and Integration for Intelligent Systems*, vol. 1, no. 1, pp. 255–260, 1999.

[13] D. I. Kosmopoulos, “Robust Jacobian matrix estimation for image-based visual servoing”, *Robotics and Computer Integrated Manufacturing*, vol. 27, no. 1, pp. 82–87, 2011.

[14] F. Chaumette and S. Hutchinson, “Visual Servo Control”, *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[15] J. Qian and J. Su, "Online estimation of image jacobian matrix by kalman-bucy filter for uncalibrated stereo vision feedback", *IEEE International Conference on Robotics and Automation*, vol. 1, no. 1, pp. 562–567, 2002.

[16] K. Stanley, Q. M. J. Wu, A. Jerbi, and W. A. Gruver, "Neural network-based vision guided robotics," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 281–286, 1999.

[17] J. Pei, S. X. Yang, and G. S. Mittal, "Vision Based Robot Control Using Position Specific Artificial Neural Network," in *International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 110–115, 2010.

[18] C. C. Cheah, C. Liu, and J. J. E. Slotine, "Adaptive Jacobian vision based control for robots with uncertain depth information," *Automatica*, vol. 46, no. 7, pp. 1228–1233, 2010.

[19] LJ Group, "http://www.ljgroup.com/", access: 7 June 2014.

[20] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons New York, 2006.

[21] Laboratory of Computational Intelligence and Robotics, "*http://ce.yazd.ac.ir/lcir/?page_id=12*", access: 21 January 2015.